DOCUMENT RESUME

ED 358 170 TM 019 974

AUTHOR Jang, Younghee
TITLE The Influence of Programming Skills on Learning and Study Strategies.
PUB DATE Apr 93
NOTE 42p.; Paper presented at the Annual Meeting of the American Educational Research Association (Atlanta, GA, April 12-16, 1993).
PUB TYPE Reports - Research/Technical (143) -- Speeches/Conference Papers (150)

EDRS PRICE MF01/PC02 Plus Postage.
DESCRIPTORS Anxiety; Calculus; *College Students; Comparative Testing; Control Groups; Experimental Groups; Higher Education; Learning Strategies; *Mathematics Achievement; Problem Solving; *Programing; Student Attitudes; Student Motivation; Study Habits; *Study Skills; *Transfer of Training
IDENTIFIERS Concentration; *PASCAL Programing Language

ABSTRACT

The relationship between learning to program at a construct level and learning and study strategies was studied for college students enrolled in a beginning Pascal programing course and a calculus course (four sections of the programing course and two of the mathematics course). For both the experimental group (n=42) and the first control group (n=51), the programing course consisted of 150 minutes of lecture and 150 minutes of laboratory each week. Both groups were given instruction in programing and Pascal, but only the experimental group was taught the loop and nested loop constructs and the corresponding Pascal codes whose outputs embodied the combination and intersection schema of Piaget. The second control group (n=38) was given instruction in calculus with 200 minutes of lecture per week for 7 weeks. The 77-item Learning Strategies and Study Skills Test, a transfer test, and an achievement test were used. Results show significant effects of learning and study strategies on the learning of complex programing constructs. A greater degree of programing skills increased the capability of selecting main ideas, but had no effect on transfer of programing skills to solving analogous problems in other domains. Results also indicate that learning complex programing constructs led to greater levels of anxiety, but had no effect on other affective factors of motivation, attitude, and concentration. Six tables present study findings. Contains 33 references. (SLD)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

ED358170

U.S. DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)
☑ This document has been reproduced as received from the person or organization originating it.
☐ Minor changes have been made to improve reproduction quality.
• Points of view or opinions stated in this document do not necessarily represent official OERI position or policy.

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY
YOUNGHEE JANG
TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

The Influence of Programming Skills on Learning and Study Strategies

by

Younghee Jang

Southwest Regional Laboratory

Los Alamitos, CA

Paper presented at the 1993 National Meeting of the American Educational Research Association, Atlanta

TM019774

ABSTRACT

This study investigated the relationship between learning to program at a construct level and learning and study strategies. Subjects were students in a four year college enrolled in the beginning Pascal programming course and a calculus course. Four sections of the programming course and two sections of the mathematics course were selected to serve either as experimental group, control group one, or control group two. For both the experimental group (n=42) and the first control group (n=51), the programming course consisted of 150 minutes of lecture and 150 minutes of laboratory per week. Both groups were given instruction in programming and in the language Pascal through lectures and laboratory activities lasting 6 weeks and 4 weeks respectively. Only the experimental group was taught the loop and nested loop constructs and the corresponding Pascal codes whose outputs embodied the combination and intersection schema of Piaget. The second control group (n=38) was given instruction in calculus consisting of 200 minutes of lectures per week for seven weeks. Results showed significant effects of learning and study strategies on the learning of complex programming constructs and that a greater degree of programming skills increased the capability of selecting main ideas but had no effect on transfer of programming skills to solving analogous problems in other domains. Results also indicated that learning complex programming constructs led to greater levels of anxiety but had no effect on other affective factors: motivation, attitude, and concentration.

INTRODUCTION

Technologists and educators believe that future classrooms will be populated with computers. Either as a medium to store and transmit knowledge or as a vehicle to enhance mental abilities, the computer is considered vital to the education of our next generation. Cognitive benefits of learning to program have been claimed by proponents of computer education (Papert, 1980), and are often used as the justification for teaching programming in our schools.

Proponents claim that learning to program will help our students learn to think more creatively and become better problem solvers. Papert (1980) believes that, when children are allowed to write programs and explore in a LOGO environment, "powerful intellectual skills are developed in the process." Bork (1981) expounds the view that analytical thinking skills applicable to a great number of areas can be learned in the context of computer programming.

There is some theoretical and logical support for this view. Problems in many different domains require substantial planning and exploration before a solution becomes apparent. In implementing a solution, the problem solver is often required to transform the problem into a suitable representation, which is often symbolic and mathematical. The solution is then carried out in an algorithmic fashion. Programming often involves planning, revision of plans, debugging, choice of representation, and algorithmic thinking. It is these characteristics of problem-solving found in programming activities that provide the basis for many of the claims.

EMPIRICAL STUDIES OF COGNITION AND PROGRAMMING

Empirical studies into the cognitive benefits of programming are of more recent origin. Since there is not a large body of codified work, it is difficult to view the studies with great specificity. Generally, the studies can be classified into three categories. The studies in the first category were empirical studies of classroom instruction of LOGO or BASIC over a fixed period of instruction. Generalized cognitive benefits were measured through correlational methods. Most of the studies in this group came earlier chronologically (Howes, O'Shea, & Place, 1980; Soloway, Lochhead, & Clement, 1982; Hines, 1983; Clements & Gullo, 1984; Milojkovic, 1984; Miller, Kelley & Kelley, 1988). The results were mixed, and there were methodological questions raising doubts about the results. However, the overall indication is that positive results are possible under certain restricted conditions, such as under well-taught and structured instruction. Pea and Kurland (1984) provided a detailed analysis of a general framework for understanding studies on the cognitive effects of learning computer programming. Part of their conclusion was that the cognitive consequences of different levels of programming skill would provide far more relevant information for guiding the process of education than standard correlation studies.

Methodological concerns such as the failure to determine the degree of learning and to relate what was learned to what was transferred resulted in another group of studies. The primary focus of these studies was not necessarily in demonstrating the cognitive benefits of programming but rather tried to achieve a better understanding of programming skills or use a more specific measure of cognitive benefit.

These studies (Clement, Kurland, Mawby, & Pea, 1986; Kurland, Pea, Clement, & Mawby, 1986; Natasi, Clements, & Battista, 1990) were more focused on particular aspects of programming. Clement et al 's study (1986) predicted and obtained a

significant correlation from pretests of analogical reasoning to posttests of programming ability to write subprocedures usable for several different programs. In Kurland et al's study (1986), programming experience (as opposed to expertise) did not appear to transfer to other domains which shared analogous formal properties. The failure to transfer could be due to the rudimentary understanding of programming. This study indicates the importance of the determination of the level of programming skill as part of any transfer study.

Cognitive Model Based Studies

Recent studies (Carver, 1988; Klahr & Carver, 1988; Dyck & Mayer, 1989; McGrath, 1988; Clements, 1990; Clements, 1991) were based on detailed models with tighter control over the instructional methods. They were either influenced by or had taken into account some of the studies on cognitive transfer in the area of cognitive psychology (Ellis, 1985; Gick & Holyoak, 1983; Holyoak, 1985; Sternberg, 1985). Some of these studies asked more specific questions. The results were sharper but of less general applicability. Klahr and Carver (1988) addressed an issue generally not dealt with in the experimental studies on programming, namely that effective transfer of knowledge depends on its degree of learning. They used an explicit model of the debugging process and showed that students who acquired better debugging strategies performed better on transfer tests of debugging in nonprogramming contexts. This experiment can be viewed as teaching a metacognitive debugging strategy which has wide general applicability across different domains.

Programming Construct Based Studies

However, none of these studies were focused at the level of specific programming constructs. If specificity in addressing what is learned to what is transferred is desired,

then a study cannot be based on an entire segment of instruction. It must be refined to a level where the programming concepts or constructs taught can be identified. A fundamental result in programming methodology is that all structured programs can be expressed in terms of three basic programming constructs: sequencing, if-then-else, and the while-loop (Boehm & Jacopini, 1966). Sequencing is simply doing one instruction after another in temporal order. Because of its perceived simplicity and generality, there has been no study conducted on its cognitive effect. Studies on the other two constructs are therefore of great significance. However there has only one such study.

This one study was on the if-then-else construct. It was conducted by Seidman (1981), and analyzed in Seidman (1989). This pioneering study was on the cognitive effects of the *if-then* and *if-then-else* statements on logical reasoning. Seidman postulated that if transfer of learning occurred and subjects interpreted the ordinary-language conditional statement in the material conditional manner, then a negative effect would be expected. The result showed a negative effect for the experimental group on reasoning with the ordinary material conditional. As pointed out by Seidman, the ordinary material conditional does not have the same semantics of the conditional *if-then* of programming languages. This study points out the importance of using analogous problems in other domains in a transfer study as unintended cognitive effects may result in transfer to similar problems with different underlying structures.

Another study focused on the programming construct level was carried out by Jang (1992) who investigated the cognitive benefits of learning to program by determining the degree of cognitive transfer of programming skills to solving analogous problems in other domains. The study focused on a programming construct, the nested loop, and investigated whether a greater degree of learning of the programming skill would lead to a greater degree of transfer to analogous problems in other domains. It further investigated whether a greater degree of transfer would result upon the receipt of a hint. The results

were positive. The study suggests that it is important to teach metacognitive strategies and applicability of programming knowledge to other domains while teaching programming.

Social and Motivational Aspects of Programming

It should also be noted that some of the investigators were also interested in the social and motivational aspects of computer programming or the computer as a medium. Milojkovic (1984) studied "the cognitive and motivational impact of exposure to microcomputers in the context of a 10-week Introduction to Problem Solving course offered to the entire fifth grade (n = 58) of a middle school in Menlo Park" (p. 25). The children were assigned randomly into 4 groups - LOGO programming, BASIC programming, CAL (Computer-Assisted-Learning), and PTP (Production Thinking Program). A number of cognitive tests were used. Milojkovic found no significant differences among all four groups. It is not clear what Milojkovic was controlling, and what were his hypotheses. Milojkovic in fact questioned the sufficiency of the amount of instruction in LOGO and BASIC in his study as to yield any cognitive benefits. His study ended up to be one measuring the effectiveness of different media and curriculum materials in problem-solving instruction, and found that there were not significant differences.

Some recent studies have focused on the environment of learning, especially in regard to fostering what Salomon and Perkins termed the "high road transfer mechanism" in the subjects. Natasi, Clements and Battista (1990) reported on their continuing work (Clements & Natasi, 1985, 1988) in the effects of LOGO programming and CAI problem-solving environments on the subjects' social-cognitive interactions, motivation and cognitive growth. While this work is not directly relevant to cognitive transfer, one should note that "children in LOGO evinced more cognitively oriented conflict, attempts at and successful resolution of conflicts, rule making, and pleasure of discovery. Resolution of cognitive conflict more than the occurrence of conflict predicted problem-solving

performance and accounted for treatment effects on problem-solving. Findings suggest that LOGO may foster cognitive growth through opportunities for resolving cognitive conflict and may enhance effectance motivation" (p. 150). This points out that even positive results in transfer studies may be due to a general improvement in the cognitive conflict resolution capability of the subjects. The confounding of the myriad of factors in a computing environment render most of these empirical studies nonreplicable. Hence it argues for transfer studies to be even more specific in relating what is learned to what is transferred.

Conclusion of Empirical Studies on Programming

The available evidence seems to show that learning to program results in *weak*, if any, transfer of cognitive skills to other domains. However, the lack of positive results on cognitive transfer in some of the studies on programming could be due to the design of the experiments being too general to detect specific transfer or due to a lack of mastery of the initial learning. In addition, it could also be due to a lack of cues as the work of Gick and Holyoak (1980, 1983) showed that both adults and children have difficulty in solving analogous problems, but if the subjects are cued or the context is made more explicit, transfer to analogous problems does occur.

These studies show that from an educational point of view, if one of the goals is to maximize students' problem-solving ability, then one must be more aware of the conditions under which specific transfer will occur. One can then devise better instructional methods and improve the learning materials. From an experimental point of view, tighter design and tests for degree of learning must be part of the experimental design as exemplified by the Klahr and Carver study and the Jang study.

Objective of Study

The objective of the present study was to investigate the influence of learning to program on learning and study strategies. The study addressed the following questions: (a) Do the procedural thinking skills in programming constructs and activities transfer to learning and study strategies ? (b) Do learning and study strategies have any effect on the learning of programming skills? (c) Does a greater degree of acquired learning and study strategies lead to a greater degree of transfer of programming skills to analogous problems in other domains ? and (d) Does learning complex programming constructs generate anxiety and impact on motivation, concentration, and attitude?

The present study differed from the methodology generally employed in current research into the cognitive benefits of computer programming in its approach and in its specificity. It focused on a specific and important construct of computer programming, namely that of the nested loop. The effect of a single programming construct on learning and study strategies was studied, rather than the generalized effects of an entire course of instruction where the relation of the materials learned to the strategies may not be entirely clear. Learning and study strategies were defined broadly by adopting a statistically valid and reliable tool for the diagnosis of study skills, the Learning and Study Strategies Inventory (LASSI). LASSI diagnoses student learning and study strategies on ten scales: Attitude, Motivation, Time Management, Anxiety, Concentration, Information Processing, Selecting Main Ideas, Study Aids, Self Testing, and Test Strategies.

Strategic skills often involve analysis, planning, implementation, monitoring and modification. In a beginning course on programming, the most complex activities are those based on the nested loop where searching and sorting algorithms are learned. Many of the aforementioned strategic skills are inherent in the teaching and learning of the nested loops and their applications. The experiment was carefully designed and the instructional phases were clearly separated out into an initial simpler phase and another phase involving the

nested loops. None of the previous studies on learning and study strategies addressed the process of learning to program with such specificity. Since the experimental results on generalized cognitive benefits were mixed, the present study tried to address some of these difficulties by focusing on an important programming construct and tried to provide a more conclusive answer to the question of effects of learning computer programming. The specificity in the choice of the programming constructs and their applications allows replication and further modification in future studies. On the other hand, a broad range of study and learning skills is included.

Intersection Schema and Combination Schema

The problems used in this study were from a set of combination and permutation problems used by Piaget and Inhelder in their studies with children's formal operational ability. The tasks studied by Piaget and Inhelder involved the generation of pairs from sets of items, for example, one version was on mixing of chemicals.

In forming all possible pairs from two sets *A* and *B*, the abstract mathematical notion Cartesian product, *A* X *B*, is the set consisting of all ordered pairs (x,y), where x is from *A* and y is from *B*. There are many different ways of forming all these pairs. Piaget envisioned the matrix schema, with the elements of *A* along one dimension and the elements of *B* along the other in a matrix, and the ordered pairs listed in a tabular form. A person with the matrix conception can systematically list all the possible pairs, for example by going along each row until the pairs are exhausted, and then continuing with the next row. This planful behavior of listing the pairs row by row is referred to as the *combination procedure schema*. More complex isomorphs exist between multiple nested loop and the Cartesian product of a collection of sets, which were also used in the present study.

In the case of the two sets being the same, as in many of the tasks (e.g., Piaget's chemical mixing task) where the order of the entries in a pair is immaterial, the *intersection procedure schema* (for a set A = {1,2,3,4,5}) is as follows:

	1	2	3	4	5
1		(1,2) ---->	(1,3) ---->	(1,4) ---->	(1,5)
2		---->	(2,3) ---->	(2,4) ---->	(2,5)
3			---->	(3,4) ---->	(3,5)
4				• ----->	(4,5)
5					

According to Piaget, children with this matrix conception could exhaust all the combinations by choosing pairs planfully from beginning to end. Children without the matrix conception typically would make pairs without an orderly pattern, and hence could not be certain of exhausting all possible pairs.

The Loop Construct

The combination and intersection schema are embedded procedurally in one of the central programming constructs, namely that of the nested iterative loop, which is used in numerous sorting and searching algorithms. Prototypic nested loops, in Pascal, are the following:

```
for I := 1 to M do                for I := 1 to N do
    for J := I to N do                for J := I+1 to N do
        Action(I,J);                      Action(I,J);
```

whose outputs are isomorphic to the combination procedure schema and intersection procedure schema respectively.

The outputs of these programs (with N = 4, and *Action(I,J)* being **begin** *write(I,J); writeln* **end**) are

```
11  12  13  14        12  13  14
21  22  23  24        23  24
31  32  33  34        34
41  42  43  44
```

respectively. In the first case, the static output is the result of a procedural manifestation of the Cartesian product of $A \times A$, where A is the set 1, 2, 3 and 4. Procedurally, the first problem is isomorphic to the combination procedure of Piaget, and the second problem is isomorphic to the intersection procedure of Piaget.

After learning the underlying control schemas, algorithms for searching and sorting were covered. In instantiating the Action(I,J) portion of the schemas in different algorithms, students were taught analysis, planning, implementation and modification of code in the context of computer programming, and these are important and complex skills. This represents a critical phase of the instruction, as the instruction and learning differ from the beginning phase of the course in degrees of difficulty significantly. In the process of acquiring an understanding of this construct, students were faced with a multitude of problems such as managing their time in their laboratory activities and understanding the concept of modular decomposition of a problem task into subproblems at a complexity level not encountered before. Therefore it may result in some changes in their attitude or learning and study strategies. The effects of learning this phase on attitude, motivation, time management, anxiety, concentration, information processing, selecting main ideas, study aids, self testing, and test strategies as measured by LASSI were studied.

METHOD

Subjects

Subjects were students in a four year college enrolled in the beginning Pascal programming course and a calculus course. Four sections of the programming course and two sections of the mathematics course were selected to serve either as experimental group, control group one, or control group two. Forty-two students in the experimental group, 51

in the first control group, and 38 in the second control group had scores on all measures administered. These subjects were from 21 different majors.

Design.

The design consists of three groups, each comprised of two sections. Four experienced computer science and mathematics instructors participated in this study. One instructor taught both sections of the experimental group, two instructors taught each section of the control group one, and the remaining instructor taught both sections of control group two. Two of the four Pascal sections were randomly assigned as the experimental group and the other two as control group one. The two calculus sections were assigned as control group two. The first control group was used to rule out the possible contribution to cognitive transfer from the initial phase of instruction in programming up to but not including the loop instruction. The second control group was used to control for self-selection factors into a programming class.

Measures

The measures administered in this study were the Learning Strategies and Study Skills Test, the Transfer Test (used as a pretest and posttest for assessing transfer), and the Achievement Test (used to assess the learning of the nested loop construct) . Learning and Study Strategies Inventory (LASSI) (1987, H&H Publishing Company, Inc.) was used to assess student's learning and study strategies and method. The LASSI focuses on both covert and overt thought and behaviors that are related to successful learning. Therefore, both student thought processes and behaviors are assessed. The LASSI consists of 77 items related to learning strategies and studying with 10 scales. These 10 scales are Attitude (ATT), Motivation (MOT), Time Management (TMT), Anxiety, (ANX) Concentration (CON), Information Processing (INP), Selecting Main Ideas (SMI), Study

Aids (STA), Self Testing (SFT), and Test Strategies (TST). Each scale has 8 items except the scale of SMI which has 5 items.

The first scale, Attitude, consists of items dealing with attitude and interest in college. The scores on this scale indicate students' general attitudes and motivation for succeeding in school and performing the tasks related to school success.

Sample Items: I feel confused and undecided as to what my educational goals should be.
I only study the subjects I like.

The second scale, Motivation, has items measuring students' diligence, self-discipline and willingness to work hard. The scores on this scale indicate the degree to which students accept responsibility for doing the tasks related to school success.

Sample Items: When work is difficult I either give up or study only the easy parts.
I set high standards for myself in school.

The scale of Time Management examines use of time management principles for academic tasks. The scores on this scale indicate the degree to which students create and use schedules.

Sample Items: I only study when there is the pressure of a test.
When I decide to study, I set aside a specific length of time and I stick with it.

The Anxiety scale has items assessing the degree to which students worry about school and their performance. The scores on this scale indicate how tense or anxious students are when approaching academic tasks.

Sample Items: Worrying about doing poorly interferes with my concentration on tests.
I am very tense when I study.

The Concentration scale deals with students' ability to pay attention to tasks. The scores on this scale indicate students' abilities to concentrate and direct their attention to school-related tasks.

Sample Items: I concentrate fully when studying.

I find that during lectures I think of other things and don't really listen to what is being said.

The Information Processing scale consists of items examining the use of imaginal and verbal elaboration, comprehension monitoring, and reasoning. The scores on this scale indicate how well students can create imaginal and verbal elaborations and organizations to enhance understanding and recall.

Sample Items: I translate what I am studying into my own words.
I try to think through a topic and decide what I am supposed to learn from it rather than just read it over while studying.

The Selecting Main Ideas scale has items addressing students' ability to select important information. The scores on this scale indicate students' skills at selecting important information on which to concentrate for further study or autonomous learning situations.

Sample Items: I have difficulty identifying the important points in my reading.
Often when studying I seem to get lost in details and "can't see the forest for the trees.

The Study Aids scale examines the degree to which students use support techniques or materials to help them learn and remember new information. The scores on this scale indicate students' ability to use and create study aids that support and increase meaningful learning and retention.

Sample Items: I use special helps, such as italics and headings, that are in my textbooks.
When they are available, I attend group review sessions.

The items on Self Testing focus on reviewing and preparing for classes and tests. The scores on this scale indicate students' awareness of the importance of self testing and reviewing (comprehension monitoring) and the degree to which they use these methods.

Sample Items: I stop periodically while reading and mentally go over or review what was said.
I try to identify potential test questions when reviewing my class material.

The last scale, Test Strategies, concentrates on students' approach to preparing for and taking tests. The scores on this last scale indicate students' use of test-taking and test preparation strategies.

Sample Items: I have difficulty adapting my studying to different types of courses.
In taking tests, writing themes, etc., I find I have misunderstood what is wanted and lose points because of it.

The transfer test was used to determine the degree of transfer from the programming domain to different domains. Two sets of questions, Form A and Form B, were constructed for use in the pretest and posttest to render them less readily recognized as identical tests. Each form contained three problems analogous to the nested loop constructs in nonprogramming domains. Both forms contained isomorphic problems with different embedding story situations. Two of the three problems dealt with the use of the intersection schema to solve the problem, and the remaining one dealt with the combination schema.

The achievement test was used to determine the subject's understanding and mastery of the nested loop concept which assessed the subjects' learning in four areas/concepts: single loop construct, intersection schema, combination schema, and nested loop construct.

The first part of the achievement test contained three program comprehension problems to assess the understanding of the codes involving the nested loop construct. For each problem, subjects were asked to write down the output of the given segment of code. The execution of the nested loops in these problems is the intersection schema and the combination schema by which the transfer tasks can be solved. The second, third, and forth parts of the test were designed to assess the ability of the subjects to write correct Pascal code involving the loop construct to generate given outputs. The second part consisted of two problems. The first problem was a simple test to see whether the subject knew the basic loop construct. The second problem tested whether the subject could apply

his or her knowledge in problem 1 to produce a given output. The third part consisted of two questions. The first was similar to the first problem of part two except it was slightly more complex. The second problem was designed to test mastery and skillful application of the nested loop to generate a given pattern. The problem was judged to be quite difficult for most subjects. A subject who solved this problem certainly had mastered the nested loop construct. The fourth part consisted of two problems. The first and second problem asked for the code to generate given outputs which embodied the intersection and combination schema respectively. Table 1 presents representative excerpts from the achievement test.

Insert Table 1 about here

Treatment

For both the experimental group and the first control group, the beginning Pascal programming course consisted of 150 minutes of classroom lecture and a 150 minutes of laboratory session each week. Each student had his or her individual terminal for laboratory work. The instruction was coordinated and uniform across all sections. All the sections followed the same instructional schedule, used the same textbook, and covered the same content. The text was *Oh! Pascal* by Doug Cooper and Michael Clancy, published by W. W. Norton & Company. The same set of laboratory activities was also used in all laboratory sessions. All the students in the experimental group and the first control group were given instruction in programming and in the language Pascal through lectures and laboratory activities.

Control Group One Introductory concepts of programming such as variables, input, output, and assignment statements were taught prior to the loop construct. A simple

treatment of functions and procedures was also given prior to the loop construct. These topics lasted until the 4th week of the instruction for the first control group.

In the first week, a brief history of computers was followed by an introduction to the computing facilities and the basic commands of the operating system, including how to edit and compile a program. Instruction on programming and Pascal began in the second week by concentrating on programs with simple input and output commands. Towards the end of the second week, the concepts of variables and debugging were covered. In the third week, expressions and assignment statements were the central topics. Progressively more complex programs were presented to illustrate these new topics. Toward the end of the third and in the fourth week, programming with procedures and function subprograms were covered. In addition, top down design and stepwise refinement were introduced. The use of procedures and functions to support the programming methodology was discussed. More advanced concepts such as the scope of a variable, value parameters and variable parameters were covered in the fourth week.

The laboratory sessions paralleled the classroom lectures. In the first week, students were taught how to log onto their computer accounts, and execute simple system commands, such as how to send and receive electronic mail, how to save and retrieve files. In the second week, students were asked to debug simple Pascal programs. They were asked to find and correct the syntax errors in the given program. In the third week, the laboratory assignment was to write a program to solve a numerical problem involving arithmetic expressions and assignment statements. In the fourth week, a more difficult program involving addition, subtraction, multiplication and division of rational numbers was assigned. In order to write this program, a function subprogram which computed the greatest common factor of two numbers was given to the student. In this lab, students became more familiar with the use of functions and procedures. In general, the laboratory assignments lagged behind the lecture to allow time for the students to absorb the concepts.

Experimental Group The instructional program was the same as the first control group during the first four weeks. The experimental group received instructions on the loop construct in the 5th and 6th weeks of instruction. The single loop construct was introduced using the *for* statement. This was followed by instruction and examples on the nested loop construct. The students were taught codes involving the nested loop construct whose outputs embodied the intersection and combination schema. Students wrote codes for problems involving the loop construct in their programming activities in the laboratory sessions.

In the fifth week, the loop construct was introduced as a mechanism for repetition. This was followed by an explanation of the syntax diagram of the **for** statement. Additional examples of the single **for** loop were given. The primary example was to introduce the iteration scheme via the example of finding the sum of the first 100 positive integers. The nested **for** loop was introduced following the single loop instruction. The fact that the inner loop is to be executed entirely before the outer loop counter variable could be changed to a new value was emphasized. The concept of a one-dimensional array was briefly discussed, as it was used in some of the lecture examples. The instructional examples were all chosen from the textbook. The application of the nested loop construct to sorting was mentioned without going into the detailed working of the program. Use of the nested loops was illustrated by many examples, including an elementary sorting method. The different forms of the nested loop used in these examples embodied both the combination and the intersection schema.

The lecture treatment concluded with instructional materials on the structure of loops and the values of the counter variables. A table of M rows and N columns can be indexed by two variables, I and J. The instructor explained that the indexing would be useful in the study of two dimensional arrays. A systematic way of going through all the entries of a table can be represented by a nested loop construct. Assignments embodying

the combination and intersection schema were given to the students to be completed in the laboratory session. Students were asked to write complete Pascal programs to generate the same output as those displayed on paper. This constituted the treatment in the fifth and sixth weeks of instruction.

Control Group Two The calculus course consisted of 200 minutes of lecture per week. The instruction was uniform across both sections. All the sections in the second semester calculus course used the same text, and covered the same topics. The text was *Calculus, Third Edition* by Howard Anton, published by John Wiley & Sons.

The instructional program for the seven-week duration followed closely the text used in the course. Both sections were taught by the same instructor. Students in the second control group had finished one semester of calculus, which covered the basic concepts of analytical geometry, and differential and integral calculus. Applications of the differential calculus were also a part of the first course. Hence instruction started with a review of the basic properties of the definite and indefinite integral. In the second week, integrals involving the trigonometric functions were covered. In the third week, the definition of the natural logarithmic function as the integral of the function 1/t from 1 to x, and its properties were covered. In the fourth week, the exponential function was introduced as the inverse of the natural logarithmic function. In the fifth week, applications of integrals of these functions to physics and economic problems were covered. In the sixth and seventh weeks, techniques of integration such as substitution, integration by parts, and methods of partial fractions were introduced. Hence the treatment of the second control group was primarily on specific functions, such as the logarithmic and exponential functions, and techniques of evaluating integrals of these two functions together with the more common polynomial and trigonometric functions.

Procedure

The study was conducted in the classes during regularly scheduled hours. Students in the programming and the mathematics sections were informed by the instructors that they were participating in a study but not informed about the nature of the study. Students were asked to attend all the classes. All the tests were administered by the instructors. For the programming sections, the test was administered at the beginning of the laboratory period; for the mathematics sections at the beginning of a class period. The transfer tests and the LASSI were not announced to the students prior to administration; the achievement test was.

Administration of premeasure In the second week of instruction prior to the instruction on Pascal, subjects in four sections of programming course and two sections of mathematics course were given the LASSI after the completion of the written transfer test. The transfer test lasted 30 minutes, and subjects took 15 to 20 minutes to complete the LASSI. After the completion of the tests, the instructor of each section continued with the normal activities of that day.

Administration of postmeasure For the postmeasures, the subjects in all three groups were given 30 minutes to complete the transfer test. After the completion of the transfer test, LASSI was administered to the subjects. The tests were completed by the subjects in one single laboratory or class period. The instructor of each section continued with the normal activities of that day after the completion of the tests.

Experimental group In the eighth week of instruction, one week after the achievement test which was used to assess subjects' understanding and mastery of the nested loop construct, the subjects were given the LASSI after the completion of the transfer test. The subjects took 15 to 20 minutes to complete the LASSI.

Control Group One In the fourth week of instruction, just prior to the students' entry into the instructional phase where the loop construct was covered, the subjects in the

first control group were given the transfer test and the LASSI which subjects took about 15 to 20 minutes to complete.

Control Group Two During the eighth week of instruction, subjects in the second control group were given the transfer test and the LASSI. The subjects took about 15 to 20 minutes to complete the LASSI.

Scoring of Learning and Study Strategies Inventory (LASSI)

In completing the LASSI the subjects read each statement and then marked their response to a key that corresponded to how well the statement described them. There were five keys, indicated by the letters "a" to "e": *not at all typical* of me, *not very typical* of me, *somewhat typical* of me, *fairly typical* of me, and *very much typical* of me.

For the scoring of the LASSI the subjects' responses were added for each scale. The score for each statement ranged from 1 to 5. Approximately half of the statements were numbered 1 to 5, and the other half were numbered 5 to 1 since some statements were stated in a positive direction and others were stated in a negative direction. There were 10 scales with each scale consisting of 8 statements except the scale of *Selecting Main Ideas (SMI)* which had 5 statements. The scales are *Attitude, Motivation, Time Management, Anxiety, Concentration, Information Processing, Selecting Main Ideas, Study Aids, Self Testing,* and *Test Strategies*. The maximum total score for each scale was 40 points except for *SMI* which was 25 points.

Scoring of Achievement Test

Total maximum score for the achievement test was 27 points. Part one, part two, and part four were given a total of 6 points respectively; part three was given a total of 9 points. Partial credit was given depending on the degree of correctness of subject's

answer. The degree of learning of the nested loop construct was given by the score of the subject on the achievement test.

Scoring of Transfer Test

The analysis of the solutions and the assignment of scores to the problems in Form A and Form B required a method for detecting the presence of the intersection procedure and the combination procedure as explained in the section on the loop construct. Each problem was given four points for a possible total of 12 points. The amount of partial credit given depended on the degree of manifestation of the intersection procedure or the combination procedure in students' solutions. There were numerous systematic or nonsystematic ways to list all the pairs or all the combinations, which yielded a complete listing of the possibilities. It should be emphasized that in this experiment one was looking for a solution which displayed the procedural manifestation of the nested loop construct, namely the intersection and combination schema, not just for any correct solution.

Intersection schema In scoring subjects' answers on the transfer test, each of the two problems which embodied the intersection procedure was given a maximum of 4 points with a total score of 8 points. Both test forms A and B for the pretest and posttest were graded in the same manner. The subject who demonstrated clear understanding of the intersection procedure whose list of pairs coincided with the order of pairs as given in the procedural manifestation of the nested loop construct was given 4 points. The amount of partial credit given depended on the degree of manifestation of the intersection procedure. This is structurally isomorphic to the output from a nested loop of the type

```
for I := 1 to 8 do
    for J := I+1 to 8 do
        writeln (I, J);
```

The order of the people in the social introduction could vary from subject to subject. Subjects could work from different arrangements of the people, for example, Nancy, Peter, Paul, Tina, Robert, Mary, Ann, and John. As long as their sequence of pairs coincided with the procedural manifestation of the nested loop construct, the solution was scored as correct.

Combination schema The solutions involving the combination schema were graded in a similar fashion to the problems involving the intersection schema. Both forms A and B in the pretest and posttest were scored in the same manner. In scoring a subject's answers, the problem which embodied the combination procedure was given a total score of 4 points. A solution list of combinations coinciding with the order in the procedural manifestation of the nested loop construct was given the maximum points. In this case, the subject clearly demonstrated understanding of the combination procedure. The amount of partial credit given depended on the degree of manifestation of the combination procedure. A correct sequence for problem 2 of Form A is

	(Salad Chicken Cake)	-> (Salad Chicken Pie)	-> (Salad Beef Cake)
->	(Salad Beef Pie)	-> (Salad Ham Cake)	-> (Salad Ham Pie)
->	(Soup Chicken Cake)	-> (Soup Chicken Pie)	-> (Soup Beef Cake)
->	(Soup Beef Pie)	-> (Soup Ham Cake)	-> (Soup Ham Pie)

which is structurally isomorphic to the output from a nested loop of the type:

```
for I:= 1 to 2 do
    for J:=1 to 3 do
        for K:= 1 to 2 do
            writeln (I,J,K);
```

with I, J and K indexing the elements of the sets {Salad, Soup}, {Chicken, Beef, Ham} and {Cake, Pie} respectively. The above sequence is a procedural manifestation of the output of the nested loop. All solutions which were permutations of the three sets {Salad, Soup}, {Chicken, Beef, Ham}, and {Cake, Pie} or permutations of the elements within each set were considered correct as long as procedurally a subject was using the combination schema from the nested loop construct. There are 6 possible permutations of the three sets, and in each permutation, there are 24 possible orderings of the elements. Hence there are altogether 144 different possible correct sequences a subject could give.

RESULTS AND DISCUSSION

This section examines the effects of the three treatment conditions - "programming and instruction on the nested loop construct": experimental group versus "programming but no instruction on the nested loop construct": control group one versus "no programming": control group two - on performance on the learning and study strategies. It also examines whether learning and study strategies have any effect on learning of the programming skills and on transfer of the programming skills to analogous problems in other domains.

To examine the initial section differences across groups, a one-way analysis of variance was performed on the scores of the transfer pretest. Results of the analysis showed no significant section differences ($F(5,125) = 1.91$, $p < .10$). As a result, the two sections in each group were combined.

Transfer of Programming Construct to Learning and Study Strategies

The question of whether the procedural thinking skills in programming constructs and activities transfer to learning and study strategies was examined. A multivariate analysis of variance (MANOVA) was used to analyze the data. The LASSI pretest,

posttest, and gain (mean difference between pretest and posttest) means and standard deviations for each group are shown in Table 2, Table 3, and Table 4. To determine whether there were any initial group differences on the LASSI pretest, a multivariate one-way analysis of variance was performed. The dependent variables were the pretest scores of the LASSI on 10 scales. Results of a multivariate tests failed to reveal significant differences among the groups ($F(20,240) = 1.37$, $p < .14$).

Insert Table 2, 3, & 4 about here

A multivariate analysis of variance (MANOVA) was performed to examine group differences for the subjects' performance on the LASSI posttest relative to their performance on the LASSI pretest. The dependent variables were the gain scores (mean difference score between posttest and pretest) of the 10 LASSI scales. Results of a MANOVA showed significant differences among the groups ($F(20, 240) = 1.80$, $p < .02$). Univariate F tests with Tukey's HSD post hoc analyses were used to determine which variables were contributing to the significance of the MANOVA. Results of univariate F tests indicated that LASSI scales which differed significantly by group were the Anxiety scale ($F(2,128) = 4.34$, $p < .02$) and the Selecting Main Ideas scale ($F(2,128) = 9.25$, $p < .0001$). Results of a Tukey's HSD post hoc analysis indicated that subjects given instruction on the nested loop construct gained significantly more on the Anxiety scale than subjects who were not taking programming (control group two). However, results of a Tukey's HSD post hoc analysis revealed that there were no significant differences between the experimental group and subjects who were taking programming but not given instruction on the nested loop construct (control group one), and between the first and second control groups.

Results of a Tukey's HSD post hoc analysis also revealed that subjects given instruction on the nested loop construct gained significantly more on the Selecting Main

Ideas scale than subjects who were taking programming but not given instruction on the nested loop construct (control group one). However, results of a Tukey's HSD post hoc analysis revealed that there was no significant differences between the experimental group and the second control group, and between the first and second control groups.

Effects of Learning and Study Strategies on Learning Programming Skills

The question of whether learning and study strategies have any effect on learning of the programming skills was examined. Scores on the six scales of the LASSI pretest which showed a significant correlation with one or both of the achievement test and the transfer posttest were added together for use in subsequent analyses. These six scales of the LASSI were Attitude, Motivation, Anxiety, Concentration, Selecting Main Ideas, and Test Strategies. Table 5 shows intercorrelations between the LASSI pretest and the achievement test, and the LASSI and the transfer posttest. The combined 6 scales of the LASSI pretest and posttest means and standard deviations for the experimental group are shown in Table 6. A multiple regression analysis was performed to examine the predictability of the LASSI for performance on the achievement test, which measured subjects' understanding and mastery of the nested loop construct.

Insert Table 5 & 6 about here

The result revealed that the combined six LASSI scales of the pretest showed a significant effect, $R^2 = .19$, $F(1,40) = 9.19$, $p < .004$. The correlation between the achievement test and the combined six scales of the LASSI pretest was .43.

Effect of Degree of Learning and Study Strategies on Degree of Transfer

The question of whether a greater degree of acquired learning and study strategies lead to a greater degree of transfer of programming skills to analogous problems in other domains was examined by using a multiple regression analysis. The posttest scores on the combined six scales of the LASSI - Attitude, Motivation, Anxiety, Concentration, Selecting Main Ideas, and Test Strategies - were used in the analysis. The scores on the LASSI were entered first and then the scores on the achievement test. Results of the analysis revealed that the LASSI posttest on the combined six scales did not show a significant effect, R^2 = .07, $F(1,40) = 3.17$, $p < .08$ whereas the achievement test showed a significant effect, change in $R^2 = .37$, $F(2,39) = 15.76$, $p < .0001$, Beta = .61. The correlation between the transfer posttest and the combined six scales of LASSI posttest was .27, and the correlation between the transfer posttest and the achievement test was .63. The partial correlation for the achievement test controlling for the combined six scales of LASSI posttest was .61 and for the combined six scales of LASSI posttest controlling for the achievement test was .24.

CONCLUSION

This study examined the effects of learning to program in a procedural language on learning and study strategies by restricting it to the micro-level of a specific and important programming construct. It also investigated the effects of learning and study strategies on learning programming. It further investigated the effects of learning and study strategies on transfer of the programming skills to solving analogous problems in other domains. It also examined whether learning complex programming constructs generates anxiety and impact on motivation, concentration, and attitude. The approach used in this study was different from the methodology generally employed in current research into the cognitive benefits of learning to program. The experiment was carefully designed to isolate the effect of a single

phase of the instructional process. The effect of an important programming construct on learning and study strategies was studied, rather than the generalized effects of an entire course of instruction. Learning and study strategies were broadly defined, and the study employed the LASSI which is a valid tool for assessing the learning strategies and study skills.

Results showed significant effects of learning the nested loop construct on the Anxiety (A) scale of LASSI for the experimental group. Subjects in the experimental group showed significant gains in the A-scale. This result was not unexpected as the design of the experiment clearly showed that the increase in anxiety is tied to the complexity of the subject matter and resulting performance.

The instruction received by the subjects in the experimental group changed in pace and complexity as the course moved into the segment on loop constructs whereas in the beginning phase of instruction, the programming skills were not of sufficient complexity to result in any changes in strategic and study skills. In the first four weeks of instruction, the materials on input, output, assignment statements, and the programs used to illustrate functions and procedures were relatively simple. Hence, the nested loop construct was the first time in the course where the students encountered substantial complexity in the subject matter. Students in the experimental group not only went through two weeks of intensive instruction on a complex construct, but also they were given an achievement test with some problems of considerable difficulty prior to their LASSI posttest. It seemed that this circumstance led to an increase in their anxiety level. Recall that LASSI was given to the first control group subjects at the end of the fourth week, just prior to instruction on the nested loop construct. This group showed almost no change on the Anxiety scale. LASSI was given to the second control group subjects at the same time the LASSI was given to the experimental group. Control group two actually showed a small decrease on the Anxiety scale. This is consistent with the self-worth theory (Covington and Beery, 1975,

Covington, 1984, 1986) on anxiety which regards anxiety reactions as caused largely by a diminution of one's sense of competency. This suggests future experiments to determine the relation of task complexity, competency, and anxiety.

Results further showed significant effect of learning the nested loop construct on the Selecting Main Ideas (SMI) scale of LASSI for the experimental group. Subjects in the experimental group showed a significant gain in the SMI-scale. There were no significant changes for the control groups. Since programming complex problems encourage a top-down approach to solving problems, the gain in the SMI-scale confirmed the transfer of this strategic skill. The lack of effects on the other scales confirmed the general results on cognitive transfer that weak or no transfer occurs except for transfers to analogous problems in other domains. However, the positive result on the SMI-scale suggests that in learning the programming construct, students abstracted important information for further use in their learning from the materials. As the ability to select important information for use in problem solving situations has a general applicability in solving problems in other domains, this result suggests that while the complex programming skills may not immediately transfer to other domains, metacognitive and learning strategies acquired by the students would improve their problem-solving abilities. This suggests the importance of future studies on the effect of learning programming on the acquisition of metacognitive skills as there has been almost no work in this area.

Concerning the effects of learning and study strategies on learning of the programming skills and construct, the subjects who scored higher on the combined scales of Attitude, Motivation, Anxiety, Concentration, Selecting Main Ideas, and Test Strategies scored significantly higher on the test of the nested loop construct. This result indicates that effective learning and study strategies have a significant influence on learning of the programming skills. Notice that all the scales aforelisted are considered to be important affective and cognitive components for learning. Subjects with high motivation for

learning coupled with effective test strategies are more likely to learn the materials in a course and do better than those without such strategies.

The results also indicated that a greater degree of learning and study strategies do not lead to a greater degree of transfer of the programming skills to solving analogous problems in other domains. This result coupled with the previous ones indicates that although the learning strategies and study skills lead to better performance on the learning of the programming skills, the transfer of the programming skills to other domains was due to the achievement of the nested loop construct.

In conclusion, this study demonstrated the effects of learning to program with specificity at the programming construct level. The division of the instructional phase into "simple" and "complex" phases was not provided in previous experimental studies. In this study, the instructional period is broken into two phases. The results on learning and study strategies showed that given sufficient complex materials in programming, improvement in specific aspects of strategic skills can occur. It also clearly demonstrated the importance of addressing factors such as anxiety while teaching programming. As a final point, the methodology of specificity at the programming construct level employed in this study holds promise for future studies and investigations in the area of procedural thinking skills, problem solving, motivation, and other affective factors.

REFERENCES

Boehm, C. & Jacopini, G. (1966). Flow diagrams, Turing machines, and languages with only two formation rules. *Communication of the Association of Computing Machinery, 9*, 5, 366-371.

Bork, A. (1981). *Learning with Computers*. Bedford, MA: Digital Press.

Carver, S.M. (1988). Learning and transfer of debugging skills: Applying task analysis to curriculum design and assessment. In R. E. Mayer (Ed.). *Teaching and learning computer programming: Multiple research perspectives* (pp. 259-298). Hillsdale: Lawrence Erlbaum Associates, Publishers.

Clement, A. C., Kurland, D. M., Mawby, R., & Pea R. D.(1986). Analogical reasoning and computer programming. *Journal of.Educational Computing Research, 2* (4), 473-486.

Clements, D. H. (1990). Metacomponential development in a LOGO programming environment. *Journal of Educational Psychology, 82* (1), 141-149.

Clements, D. H. (1991). Enhancement of creativity in computer environments. *American Educational Research Journal, 28* (1), 173-187.

Clements, D. H., & Gullo, D. F. (1984). Effects of computer programming on young children's cognition. *Journal of Educational Psychology, 76,* 1051-1058.

Clements, D. H., & Nastasi, B. K. (1985). Effects of computer environments on social-emotional development: Logo and computer-assisted instruction. *Computers in the schools, 2* (2/3), 11-31.

Clements, D. H., & Nastasi, B. K. (1988). Social and cognitive interactions in educational computer environments. *American Educational Research Journal, 25*, 87-106.

Covington, M. V. (1984). The Self-worth theory of achievement motivation: Findings and implications. *Elementary School Journal, 85*, 5-20.

Covington, M. V. (1986). Anatomy of failure-induced anxiety: The role of cognitive mediators. In R. Schwarzer (ed.), *Self-Related Cognitions in Anxiety and Motivation*. New York, Erlbaum, 247-263.

Covington, M. V., and Beery, R. (1975). *Self-worth and School Learning*. New York: Holt, Rinehart and Winston.

Dyck, J. L., & Mayer, R. E. (1989). Teaching for transfer of computer program comprehension. *Journal of Educational Psychology, 81*(1), 16-24.

Ellis, H. C. (1965). *The transfer of learning*. New York: Macmillan.

Gick, L. M., & Holyoak, K. J. (1980). Analogical solving. *Cognitive Psychology, 12*, 306-355.

Gick, L. M., & Holyoak, K. J. (1983). Schema induction and analogical transfer. *Cognitive Psychology, 15*, 1-38.

Hines, S. N. (1983, July-August). Computer programming abilities of five-year-old children. *Educational Computer*, 10-12.

Holyoak, K. J. (1985). The pragmatics of analogical transfer. In G. H. Bower (Ed.), *The psychology of learning and motivation* (Vol. 19, pp. 59-87). New York: Academic Press.

Howe, J. A. M., O'Shea, T. and Plane, F. (1980). Teaching mathematics through Logo programming: An evaluation study. In R. Lewis and E. D. Tagg (Eds.) *Computer assisted learning: Scope, progress and limits* (pp. 85-102). New York: North-Holland Publishing Company.

Inhelder, B., & Piaget, J. (1969). In E. A. Lunzer & D. Papert (Trans.), *The early growth of logic in the child: Classification and seriation*. New York:Norton.

Jang, Y. (1992, April) Cognitive transfer of computer programming skills and analogous problem solving. Paper presented at the *American Educational Research Association,* 1992, San Francisco.

Klahr, D., & Carver, M. (1988) Cognitive objectives in a LOGO debugging curriculum: Instruction, learning, and transfer. *Cognitive Psychology, 20,* 362-404.

Kurland, D. M., Pea, R. D., Clement, C., & Mawby, R. (1986). A study of the development of programming ability and thinking skills in high school. *Journal of Educational Computing Research, 2* (4), 429-458.

McGrath, D. (1988). Programming and problem solving: Will two languages do it? *Journal of Educational Computing Research, 4* (4), 467-484.

Miller, R. B., Kelly, G. N., & Kelly, J. T. (1988). Effects of LOGO computer programming experience on problem solving and spatial relations ability. *Contemporary Educational Psychology, 13,* 349-357

Milojkovic, J. D. (1984). *Children learning computer programming: Cognitive and motivational consequences.* Unpublished doctoral dissertation, Stanford University.

Nastasi, B. K., Clements, D. H., & Battista, M. T. (1990). Social-cognitive interactions, motivation, and cognitive growth in LOGO programming and CAI problem-solving environments. *Journal of Educational Psychology, 82* (1), 150-158.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas.* New York: Basic books.

Pea, R. D., & Kurland, D. M. (1984). On the cognitive effects of learning computer programming. *New Ideas in Psychology, 2,* 137-168.

Seidman, R. H. (1981, April). The effects of learning a computer programming language on the logical reasoning of school children. Paper presented at the *Annual Meeting of the American Educational Research Association* , Los Angeles, CA..

Seidman, R. H. (1989). Computer programming and logical reasoning: Unintended cognitive effects. *Journal of Educational Technology Systems, 18* (2) , 123-141.

Soloway, E., Lochhead, J., & Clement, J. (1982). Does computer programming enhance problem solving ability? Some positive evidence on algebra word problems. In R. Seidel, R. Anderson, & B. Hunter (Eds.), *Computer Literacy*. New York: Academic Press.

Sternberg, R. J. (1985). *Beyond IQ: A triarchic theory of human intelligence*. Cambridge, MA: Cambridge University Press.

Table 1 : Excerpts of Representative Problems from Achievement Test on Loop Construct

1. What is the output of each of the following segments of code?

(a)
```
for K := 1 to 5 do
    begin
        for J := K+1 to 5 do
            write (K,J);
        writeln
    end;
```

(b)
```
for I := 1 to 2 do
    for J := 3 to 4 do
        for K := 1 to 3 do
            begin
                write (I,J,K);
            writeln
            end;
```

2. Write a Pascal nested loop code segment to generate on the screen
Output on the screen is :

```
|     *
|   * * *
| * * * * *
|* * * * * * *
```

where | denotes the left edge of the screen

3. Write code to generate the following output on the screen:

```
1 6
1 7
1 8
1 9
2 7
2 8
2 9
3 8
3 9
4 9
```

Table 2 : Pretest, Posttest, and Gain Means and Standard Deviations of Experimental Group for LASSI

	Experimental Group						
	Pretest		Posttest		Gain		
Scales	M	(SD)	M	(SD)	M	(SD)	n
Attitude	32.17	(5.69)	32.36	(4.49)	.19	(4.73)	42
Motivation	30.93	(5.62)	30.12	(5.08)	-.81	(5.09)	42
Time Management	25.17	(5.13)	25.40	(5.71)	.24	(3.67)	42
Anxiety	25.93	(6.69)	27.74	(6.86)	1.81	(4.33)	42
Concentration	27.38	(5.05)	27.36	(5.23)	-.02	(4.63)	42
Information Processing	29.17	(5.50)	29.21	(5.54)	.05	(3.60)	42
Selecting Main Ideas	18.31	(3.89)	19.45	(3.18)	1.14	(2.98)	42
Study Aids	24.71	(4.67)	25.33	(5.29)	.62	(3.49)	42
Self Testing	27.64	(5.10)	27.98	(4.57)	.33	(3.86)	42
Test Strategies	30.14	(5.85)	29.69	(6.49)	-.52	(4.12)	42

The total possible scores for each scale is 40 points except for Selecting Main Ideas which is 25 points.

Table 3: Pretest, Posttest, and Gain Means and Standard Deviations of Control Group One for LASSI

	Control Group One						
	Pretest		Posttest		Gain		
Scales	M	(SD)	M	(SD)	M	(SD)	n
Attitude	32.51	(5.86)	32.16	(4.32)	-.29	(4.31)	51
Motivation	30.84	(5.62)	29.63	(4.96)	-1.14	(3.60)	51
Time Management	24.37	(5.90)	25.0	(5.36)	.63	(3.50)	51
Anxiety	27.02	(6.22)	27.06	(6.78)	.04	(3.59)	51
Concentration	26.41	(5.29)	26.65	(5.21)	.24	(3.33)	51
Information Processing	27.10	(5.94)	27.18	(5.94)	.08	(4.15)	51
Selecting Main Ideas	19.75	(2.92)	18.61	(2.82)	-1.16	(2.53)	51
Study Aids	24.10	(5.51)	24.08	(5.29)	.02	(4.92)	51
Self Testing	25.71	(6.16)	25.94	(5.21)	.26	(3.36)	51
Test Strategies	30.94	(4.80)	29.78	(4.52)	-1.20	(3.69)	51

The total possible score for each scale is 40 points except for Selecting Main Ideas which is 25 points.

Table 4: Pretest, Posttest, and Gain Means and Standard Deviations of Control Group Two for LASSI

	Control Group Two						
	Pretest		Posttest		Gain		
Scales	M	(SD)	M	(SD)	M	(SD)	n
Attitude	34.13	(3.40)	33.87	(4.19)	-.34	(3.84)	51
Motivation	33.0	(3.96)	31.68	(4.28)	-1.26	(3.29)	51
Time Management	25.95	(5.77)	25.97	(6.70)	-.18	(3.32)	51
Anxiety	26.87	(6.03)	26.45	(6.53)	-.37	(2.51)	51
Concentration	28.97	(5.07)	28.11	(5.91)	-.74	(3.25)	51
Information Processing	28.13	(4.11)	28.82	(4.48)	.90	(3.57)	51
Selecting Main Ideas	18.61	(2.66)	18.74	(2.24)	-.08	(2.07)	51
Study Aids	24.71	(5.09)	24.74	(4.12)	-.03	(3.71)	51
Self Testing	28.45	(4.82)	27.66	(4.38)	-.61	(3.15)	51
Test Strategies	31.34	(4.02)	31.18	(4.08)	-.26	(2.63)	51

The total possible score for each scale is 40 points except for Selecting Main Ideas which is 25 points.

Table 5 : Intercorrelations Between Achievement Test, Transfer Posttest, and LASSI Pretest for the Experimental Group

LASSI Scales	Transfer Posttest	Achievement Test
Attitude	.39*	.43**
Motivation	.31*	.29
Time Management	.15	.16
Anxiety	.34*	.29
Concentration	.42**	.30*
Information Processing	.15	.19
Selecting Main Ideas	.43**	.43**
Study Aids	-.19	-.19
Self Testing	.13	.04
Test Strategies	.44**	.41**

* $p < .05$. ** $p < .01$.

Table 6: Pretest and Posttest Means and Standard Deviations for The Total of the Six Scales of LASSI

	Pretest		Posttest			
	M	(SD)	M	(SD)	Max. Pts	n
Sum of 6 Scales	164.86	(26.07)	169.62	(25.12)	225	42